Predicting the Best Units within a Fleet: Prognostic Capabilities Enabled by Peer Learning, Fuzzy Similarity, and Evolutionary Design Process

Piero P. Bonissone, Anil Varma
General Electric Global Research
One Research Circle, Niskayuna, NY 12309, USA
E-Mail: {bonissone,varma}@research.ge.com

Abstract. We analyze the task of selecting the most reliable units within a fleet of vehicles and formulate it as a prediction and classification problem. The prediction of each unit's remaining life is based on the identification of "peer" units, i.e. vehicles with similar utilization and maintenance records that are expected to behave similarly to the unit under consideration. With these peers, we construct local predictive models to estimate the unit's remaining life. We use evolutionary algorithms (EA's) to develop the criteria for defining peers and the relevance of each criterion in evaluating similarity with the unit. Each individual in the EA's population fully characterizes an instance-based fuzzy model that is used to predict the unit's remaining life. The precision of the selection of units with best-expected life provides the fitness value. We analyzed the performance of the evolutionary approach over two years of operation and maintenance data for a fleet of 1100 locomotives. The results illustrate the high accuracy and robustness of this approach. In the conclusion, we highlight the implications of this approach for supporting the lifecycle of the fuzzy models.

I. Introduction

The behavior of complex electromechanical assets, such as locomotives, tanks, and aircrafts, varies considerably across different phases of their lifecycle. Assets that are identical at the time of manufacture will *'evolve'* into somewhat individual systems with unique characteristics based on their usage and maintenance history. Utilizing these assets efficiently requires a) being able to create a model characterizing their expected performance, and b) keeping this model updated as the behavior of the underlying asset changes. This paper outlines a fuzzy peer-based approach for performance modeling combined with an evolutionary framework for model maintenance. A series of experiments using data from locomotive operations were conducted and the results from this initial validation exercise are presented.

A. Problem Description

The focus of this series of experiments is mission reliability. This can be broadly defined as – given a mission of duration X days, what percentage of units assigned to that mission are able to complete the mission without a critical failure. The motivation for high mission reliability in both commercial and DOD environments is two-fold. First, it makes mission planning and execution more predictable and effective. Second, it reduces the logistics footprint required to support a certain level of readiness. In the military domain, it may mean picking the best 5 vehicles to conduct a reconnaissance mission in swampy terrain. In the commercial sector, it may imply selecting the best 5 locomotives to deliver time-critical shipments from coast to coast. This problem is accentuated in the case of new mission types or new equipment platforms when insufficient data exists on how the equipment will behave in that environment.

B. Paper Structure

In section 2, we cite related work and define the focus of our work: the use of evolutionary search for model generation and maintenance. In section 3, we describe the data used to evolve out models, and the baselines and metrics used to evaluate our experiments. In section 4, we formalize our proposed approach, the evolution of peer-based predictive models. In the last two sections, we show the results of our experiments, describe the promise of this methodology, and discuss its possible future extensions.

II. Related Work

A. Model Characterization

A model is characterized by its *representation* (structural and parametric information) and its associated *reasoning mechanism*, which is usually related to the representation. The generation and updating of a model requires a *search* method to define the model's representation and resolve any degree of freedom in designing its reasoning mechanism. This definition applies across a broad class of models, ranging from a common representation for physics-based models (Linear Differential Equations), to widely popular probabilistic models (Bayesian Belief Networks), to powerful functional approximators (Neural and Fuzzy Systems) and to non-graphical representation, such as instance-based models. This is summarized in Table I and further described in [1].

TABLE I
Representation (Structure and Parameters), Reasoning Mechanisms, and Design Search Methods for Modeling Techniques

Modeling Technique	**LDE**	**BBN**	**NN**	**TSK**	**Instance-based**
Model Structure	Order	Topo-logy	Topology	Rule Set	Attribute Space
Model Parameters	Coeffi-cients	Prior Prob. Condit. Prob.	Biases Weights	Term Sets Scaling Factors Coeffi-cients	Attribute Weights & Similarity Parameter
Reasoning Mechanism	Solve Eqs.: Closed Form or Approx	Node evaluat. & Propa-gation	Node evaluation & Propa-gation	Node evaluat. & Propa-gation	Local Model evaluation & Output Aggreg.
Design Search Method	First Princip. Energy-based Methods	Manual EA EM …	Manual EA Backprop Conjugate Grad.	Manual EA Back-prop …	Manual EA …

B. Evolutionary Approach to Designing Models

The main contribution of our methodology is the use of evolutionary search to generate instance-based predictive

models. In the past, EA's have been extensively used to evolve neural networks [2-3], Bayesian belief networks [4], fuzzy systems [5], and case-based classifiers [6]. However, to the best of our knowledge, they have never been used to design instance-based predictors.

III. Data Collection and Experiments Set-up

A. Data Sources

The data used to train our model and validate our experiments were collected from four different sources, as illustrated in Figure 1.

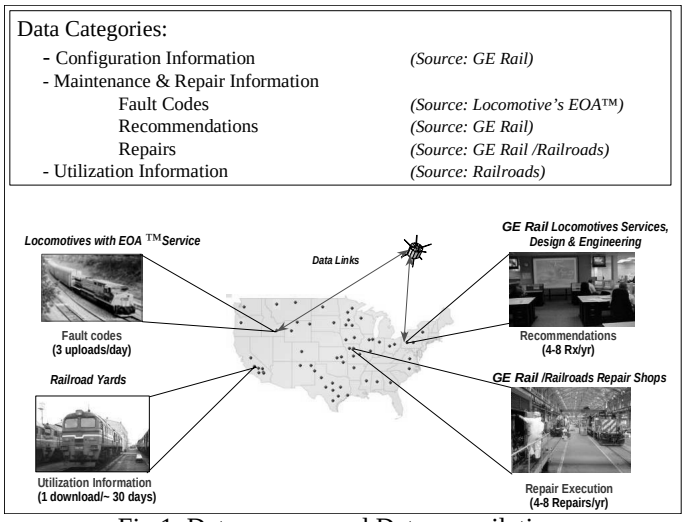

Fig 1. Data sources and Data compilation

1) Locomotive Design & Engineering Data from GE Rail: GE Rail manufactured the locomotives in this study. As the OEM, GE Rail possessed engineering data on locomotive models, configurations, date of manufacture, date of service, the date EOA service was installed, upgrades and software modifications (Figure 1 top-right corner).

2) Locomotive Recommendation Data from GE Rail EOA™ remote monitoring and diagnostics service: For each locomotive, there was a time-stamped record of when the Expert on Alert (EOA™) system detected abnormal patterns in the fault data (Figure 1 top-left corner), leading to a recommendation being issued by GE Rail Locomotive Services (Figure 1 top-right corner). A red or yellow recommendation indicated a problem that was serious and required a fix in the next 7-10 days at most.

3) Locomotive Maintenance Data from Repair Shops: Each red or yellow recommendation used in the experiments was associated with maintenance feedback from railroads or GE repair shops (Figure 1 bottom-right corner), which indicated the exact repair action that successfully fix the problem. This allowed us to screen the data and include only maintenance intervals where a genuine problem existed on the locomotive that was verified by the maintenance personnel.

4) Locomotive Utilization data from a selected railroad:

Each locomotive maintains an on-board record of a number of utilization-related parameters that are collected when a locomotive reaches a railroad yard (Figure 1 bottom-left corner). These parameters include odometer miles, total megawatt-hours, hours spent motoring, hours spent in dynamic braking, cumulative engine hours, cumulative engine hours moving, percentage of time spent in each of the eight notch settings (analogous to gear settings) and others.

B. Data Segmentation

We wanted to investigate how environmental, operational, or maintenance changes could affect our experiments and whether our learning techniques could adapt to such changes. Furthermore, we needed to assess how the incremental acquisition of data could improve the performance of our learning techniques. To this end we decided to create three data slices, on *May 22, 2002*, *November 1, 2002*, and *May 1, 2003*, respectively. The size of the fleet increased over time (from 262 to 634 and 845 units, respectively), as new units were placed in service and as we started collecting more utilization and maintenance data from previous units. As a result the number of the best 20% performers increased with the size of the fleet to 52, 127, and 169 units, respectively. This is shown at the bottom of Figure 2.

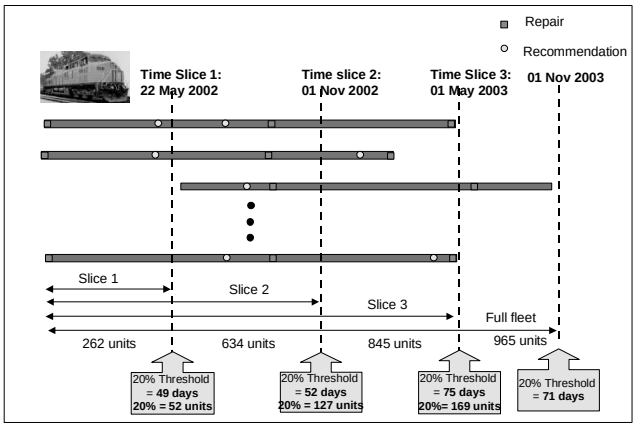

Fig 2: Data Slices used in Experiments

These three data slices were used to determine the *target* units for our selection. For each data slice we constructed three sets of targets (i.e., ground truth for the experiments): 1) the *best 20 % past performers*; 2) the *best 52 past performers;* 3) the *best 20% future performers.* We identified the first two target sets by sorting the units in decreasing order using the *median time-between-failure* of past operational durations, until we identified the best 20% or the best 52 units. We identified the last target set by sorting the units in decreasing order using the duration of the *first period of operation* after the data slice.

C. Performance Metric

The primary metric was the ability of a classifier to select the best N units in any given slice (i.e., its precision). We investigated two approaches to define the top 'N' units.

1) *Fixed Percentage Approach:* In the fixed percentage approach, the task of the classifier was to pick the top 20% of units. The actual number of units to be selected increased with fleet size as we progressed from slice 1 to slice 3.

2) *Fixed Number Approach:* In the fixed number of units approach, the actual number of units to be selected was kept constant. In the experiments, the number used was 20% of the first slice, i.e. 52 units. As the size of the fleet increased, the selection task got harder as 52 units represented 12% and 6% of the fleet size in slices 2 and 3 respectively.

D. Baselines

Two baselines were calculated to measure the increase in capability provided by the collective mind algorithms.

1) *Random*: The first baseline measured the expected performance if selection of the best *N* units were done randomly. This obviously represented a worst-case scenario.

2) *Heuristics*: The second baseline was the best performance achieved by single or multiple heuristics (based on any of the measured or calculated parameters) which we used to rank the fleet and pick the best *N* units.

IV. Proposed Learning Methodology

A. Peer-based Learning Methodology

In our experiments we wanted to explore peer-based learning methodologies, since they provide a transparent, adaptable model mechanism. We focused on the representation and reasoning mechanisms of instance-based reasoning. After developing several exploratory peer-based models, using a manual process based on WEKA [7], we decided to use a fuzzy instance-based classifier (FIBC) designed by an evolutionary search (instead of a manual process). In section IV B, we will describe the FIBC, while its EA-based design will be detailed in section IV C.

B. Fuzzy Instance-Based Classifiers (FIBC)

Instance-based reasoning (IBR) relies on a collection of previously experienced data that can be kept in their raw representation. Unlike Case-based Reasoning (CBR), they do not need to be refined, abstracted and organized as cases. Like CBR, IBR is an analogical approach to reasoning, since it relies upon finding previous instances of *similar* problems and uses them to create an ensemble of local models. Hence the definition of similarity plays a critical role in the performance of IBR's. Typically, similarity will be a dynamic concept and will change over the use of the IBR. Therefore, it is important to apply learning methodologies to define and adapt it. Furthermore, the concept of similarity is not crisply defined, creating the need to allow for some degree of vagueness in its evaluation. We addressed this issue by evolving the design of a similarity function in conjunction with the design of the attribute space in which the similarity was evaluated. Specifically we used the following four steps:

1) Retrieval of similar instances from the Data Base
2) Evaluation of similarity measure between the probe and the retrieved instances
3) Creation of local models using the most similar instances (weighted by their similarity measures)
4) Aggregation of outputs of local model to probe

1) *Retrieval*: The retrieval step consists in finding all DB instances whose behavior is similar to the probe. These instances are the probe's potential peers and can be seen as points in an *n*-dimensional feature space. For instance, let us assume that a probe Q has an associated *n*-dimensional vector of values for each potential attribute:

$$Q = [x_{1,Q}, x_{2,Q}, ..., x_{n,Q}] \quad (1)$$

A similar n-dimensional vector characterizes each unit u_i in the fleet. Furthermore, each unit has an attached vector $O(u_i)=[D_{1,i}, D_{2,i}, \ldots, D_{k(i),i}]$ containing its historic operational availability durations:

$$u_i = [x_{1,i}, x_{2,i}, ..., x_{n,i}]; O(u_i) = [D_{1,i}, D_{2,i}, ..., D_{k(i),i}] \quad (2)$$

For each dimension *i*, we define a *Truncated Generalized Bell Function*, $TGBF_i(x_i;a_i,b_i,c_i)$, centered at the value of the probe c_i, which represents the degree of similarity along that dimension. Specifically:

$$TGBF(x_i;a_i,b_i,c_i) = \begin{cases} \left[1+\left|\frac{x_i-c_i}{a_i}\right|^{2b_i}\right]^{-1} & if \left[1+\left|\frac{x_i-c_i}{a_i}\right|^{2b_i}\right]^{-1} > \varepsilon \\ 0 & otherwise \end{cases} \quad (3)$$

where ε is the truncation parameter, e.g. $\varepsilon = 10^{-5}$

Since the parameters c_i in each $TGBF_i$ are determined by the values of the probe, each $TGBF_i$ has only two free parameters, a_i and b_i, to control its spread and curvature. In a coarse retrieval step, we extract an instance in the DB if all of its features are within the ***support*** of the TGBF's, using standard SQL queries. Now we can formalize the retrieval step. *P(Q)*, the set of potential peers of *Q* to be retrieved, is composed of all units within a range from the value of Q: $P(Q) = \{u_j, j = 1,...,m \mid u_j \in N(Q)\}$ and *N(Q)* - a neighborhood of Q – is defined by the constraint $|x_{i,Q} - x_{i,j}| < R_i$ for all potential attributes *i* for which their corresponding weight is non-zero. R_i is half of the support of the $TGBF_i$, centered on the probe's coordinate $x_{i,Q}$.

2) *Similarity Evaluation*: Each $TGBF_i$ is a membership function representing the degree of satisfaction of constraint $A_i(x_i)$. Thus, $TGBF_i$ measures the *closeness* of an instance around the probe value $x_{i,Q}$ along the i^{th} attribute. For a potential peer P_j, we evaluate $S_{i,j} = TGBF(x_{i,j}; a_i, b_i, x_{i,Q})$, its similarity with the probe Q along each attribute *i*. The values (a_i, b_i) are design choices initially chosen manually, and later determined by the EA's. Since we want the most similar instances to be the closest to the probe along ***all*** *n* attributes, we use a similarity measure defined as the intersection (minimum) of the constraint-satisfaction values:

$$S_j = Min_{i=1}^{n}\{S_{i,j}\} = Min_{i=1}^{n}\{TGBF(x_i;a_i,b_i,x_{i,Q})\} \quad (4)$$

Equation (4) implies that each attribute is equally important in computing similarity. In our case, however, we consider each criterion to have a different relevance in that computation. Therefore, we attach a weight w_i to each attribute A_i and we extend the notion of similarity measure between P_j and the probe Q using a weighted minimum operator:

$$\begin{aligned} S_j &= Min_{i=1}^{n}\{Max[(1-w_i), S_{j,i}]\} \\ &= Min_{i=1}^{n}\{Max[(1-w_i), TGBF(x_i;a_i,b_i,x_{i,Q})]\} \end{aligned} \quad (5)$$

where $w_i \in [0,1]$. The set of values for the weights $\{w_i\}$ and of the parameters $\{(a_i, b_i)\}$ are critical design choices that impact the proper selection of peers. In this section we assume a manual setting of these values. In section IV C we will explain their derivation using evolutionary search.

3) *Creation of Local Models*: The idea of avoiding pre-constructed models and creating local model when needed can be traced back to memory-based approaches [8] and lazy-learning [9-10]. Within the scope of this paper, we will focus on the creation of local predictive models used to forecast each unit's remaining life. First, we will use each local model to

generate an estimated value of the predicted variable. Then, we will use an aggregation mechanism, based on the similarities of the peers, to determine the final output.

For instance, let us assume that for a given probe Q we have retrieved *m* peers, $P_i(Q)$, *i=1,…, m*. Each peer $P_i(Q)$ has a similarity measure S_i with the probe. Furthermore, each peer P_i has a track record of operational availability between failures $O(P_i) = [D_{1,i}, D_{2,i}, \ldots, D_{k(i),i}]$. Note that each peer $P_i(Q)$ will have *k(i)* availability pulses in its track history. For each peer P_i, the goal is to determine the duration of the ***next*** availability duration $D_{k(i)+1,i.}$ Then we want to combine the prediction of all the peers $\{D_{k(i)+1,i}\}$ *(i=1,…, m)* to estimate the availability duration for the probe Q.

Since we did not have enough historical data to reliably generate local regressions, we experimented with simpler models, such as averages and medians. We found that the most reliable way of generating the next availability duration $D_{k(i)+1,i.}$ from the operational availability vector $O(P_i) = [D_{1,i}, D_{2,i}, \ldots, D_{k(i),i}]$ was to use an exponential average that gives more relevance to the most recent information, namely:

$$D_{k(i)+1,i} = \overline{D}_{k(i),i} = (1-\alpha)^{k(i)-1} D_{1,i} + \sum_{j=2}^{k(i)} (1-\alpha)^{k(i)-j} \times \alpha \times D_{j,i} \quad (6)$$

Again, critical to the performance of this model is the choice of the value of parameter α. Section IV C will illustrate how to determine this value using evolutionary search.

4) *Aggregation of Local Models' Outputs*: We need to combine the individual predictions $\{D_{k(i)+1,i}\}$ *(i=1,…, m)* of the peers $P_i(Q)$ to generate the prediction of the ***next*** availability duration, $D_{Next,Q}$ for the probe Q. We define this aggregation as the *similarity weighted average*, by computing the weighted average of the peers' individual predictions using their normalized similarity to the probe as a weight:

$$D_{Next,Q} = \frac{\sum_{i=1}^{m} S_i \times D_{k(i)+1,i}}{\sum_{i=1}^{m} S_i} \quad (7)$$

Given the critical role played by the weights $\{w_i\}$, by the search parameters $\{(a_i, b_i)\}$ and by exponent α, it was necessary to create a methodology that could generate the best values according to our metrics (classification precision).

C. Evolution of Fuzzy Instance Based Classifiers (FIBC)

After testing several manually-design peer-based models, we decided to use evolutionary search to develop and maintain the fuzzy instance based classifier. Using the wrapper methodology detailed in [6] we defined the use of Evolutionary Algorithms (EA's) to tune the parameters of a classifier used to underwrite insurance applications. In this application, we extend evolutionary search beyond parametric tuning to include structural search, via attribute selection and weighting [11].

1) *EA Architecture:* The EA's are composed of a population of individuals ("chromosomes"), each of which contains a vector of elements that represent distinct tunable parameters within the FIBC configuration. Examples of tunable parameters include the range of each parameter used to retrieve neighbor instances and the relative weights associated with each parameter used for similarity calculation. The EA's used two types of mutation operators (Gaussian and uniform), and no crossover. Its population (with 100 individuals) was evolved over 200 generations.

2) *Chromosome Representation:* Each chromosome specifies a vector of weights $[w_1, w_2, \ldots w_D]$ and defines an instance of the attribute space used by its associated classifier. If $w_i \in \{0,1\}$, we perform *attribute selection*, i.e., we select a crisp subset of the universe of potential attributes. If $w_i \in [0,1]$ we perform *attribute weighting*, i.e., we define a fuzzy subset of the universe of potential attributes.

$$[w_1\ w_2 \ldots w_D][(a_1,b_1),(a_2,b_2),\ldots,(a_D,b_D)][\alpha] \quad (8)$$

where $w_i \in [0,1]$ *for* attribute *weighing* or

$w_i \in \{0,1\}$ *for* attribute *selection*

D = Cardinality of universe of features U, $D = |U|$

$d = \sum_i^D w_i$ (fuzzy) cardinality of selected features

(a_i, b_i) = Parameters for GBF_i

α = Parameter for Exponential Average

In summary, the first part of the chromosome, containing the weights vector $[w_1, w_2, \ldots w_D]$, defines the attribute space (e.g. the FIBC structure) and the relevance of each attribute in evaluating similarity. The second part of the chromosome, containing the vector of pairs $[(a_1, b_1), \ldots (a_i, b_i), \ldots (a_D, b_D)]$ defines the parameter for the retrieval and similarity evaluation. The last part of the chromosome, containing the parameter α, defines the local model.

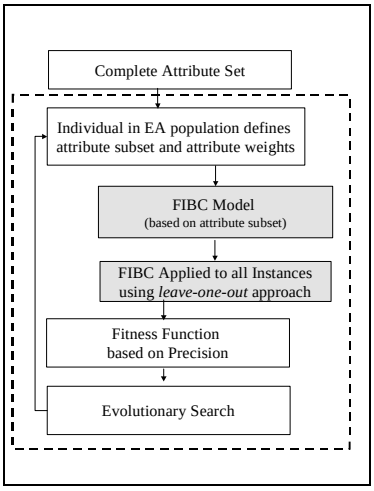

Fig 3. Fitness function using Wrapper Approach

2) *Fitness Function:* The fitness function is computed using the *wrapper* approach [11] – see figure 3. For each chromosome, represented by (8), we instantiate its corresponding FIBC. Following a *leave-one-out* approach, we use the FIBC to predict the expected life of the unit under question – see (7) – following steps 1-4, as described in section IV B. We repeat this process for all units in the fleet, and we sort them in decreasing order, using their predicted duration $D_{Next,\ Q}$. Then we select the top 20%. The fitness function of the chromosome is the precision of the classification, TP/(TP+FP).

For each set of experiments, we used the three time slices illustrated in Figure 2. Our goal was to test the adaptability of the learning techniques to environmental, operational, or

maintenance changes. Furthermore, we wanted to determine if their performance could improve over time, with incremental data acquisition.

V. RESULTS

A. First Experiment Set: Top 20% Past Performers

For each time slice, we used EA's to generate an optimized weighted subset of attributes to define the *peers* of each unit. We used the ***evolved peer*** approach to predict the ***best 20% of the fleet***, based on their past performance. In this case a random selection would yield 20%.

Time slice 1 (fleet size = 262 units; top 20% = 52 units): Evolved Peers outperformed both manually designed peers and heuristics/fleet -based approach: **48% vs. 41% vs. 32%.**

Time slice 2 (fleet size = 634 units; top 20% = 127 units): Evolved Peers outperformed both manually designed peers and heuristics/fleet -based approach: **56% vs. 55% vs. 46%.**

Time slice 3 (fleet size = 845 units; top 20% = 169 units): Evolved Peers outperformed both manually designed peers and heuristics/fleet -based approach: **60% vs. 54% vs. 50%.**

B. Second Experiment Set: Top 52 units Past Performers

Since the fleet size at each start-up time was different, we repeated the same experiments keeping the number of units constant (52 units) over the three start-up times, instead of keeping a constant top 20%. Thus the random selection baseline changed from [20%-20%-20%] to [20%-8%-6%], i.e., 52/262=20%; 52/634=8%; 52/845 = 6%. Furthermore, given the superior performance of the evolved peers over the manually constructed peers, we decided to use the optimized peer design instead of the manual one in all subsequent experiments. The following results are shown in Figure 4.

Time slice 1 (fleet size = 262 units; top 52 units= 20%): Evolved Peers outperformed heuristic/fleet -based approach: **48% vs. 32% (vs. 20%** random selection).

Time slice 2 (fleet size = 634 units; top 52 units = 8%): Evolved Peers outperformed heuristic/fleet -based approach: **56% vs. 37% (vs. 8%** random selection).

Time slice 3 (fleet size = 845 units; top 52 units = 6%): Evolved Peers outperformed heuristic/fleet -based approach: **63.5% vs. 37% (vs. 6%** random selection).

C. Third Experiment Set: Top 20% Future Performers

In the previous two sets of experiments, the targets for selection were the best-known units in the fleet based on their past performance. After these experiments were completed in April, we switched the target of the selection to the one with the ***best next*** pulse duration. Once again, the same methodology, based on evolved peers outperformed the rest – see Fig 5. In this case a random selection would yield 20%.

Time slice 1 (fleet size = 262 units; top 20% = 52 units): Evolved Peers outperformed both heuristic/fleet and unit's own history-based approach: **43% vs. 32% vs. 26.5%.**

Time slice 2 (fleet size = 634 units; top 20% = 127 units): Evolved Peers outperformed both heuristic/fleet and unit's own history-based approach: **42.2% vs. 42% vs. 33.6%.**

Time slice 3 (fleet size = 845 units; top 20% = 169 units): Evolved Peers outperformed both heuristic/fleet and unit's own history-based approach: **54% vs. 36% vs. 33.8%.**

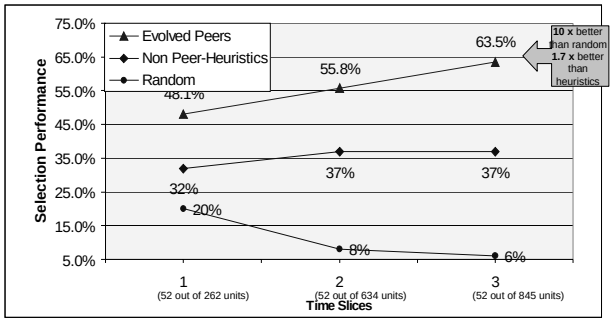

Fig. 4. 2nd Set of Experiments (52 Units; Past Performers)

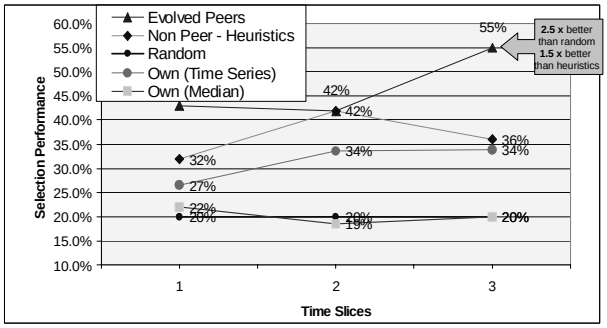

Fig. 5. 3rd Set of Experiments (20% - Future Performers)

VI. CONCLUSIONS

A. Manually Designed versus Evolved Peer-based Models

The static (manually designed) peers outperformed both single and multiple heuristics based on fleet-wide data. They exhibited a graceful degradation when critical information was not provided (robustness to information loss). We performed an experiment comparing the manually designed peers with the best heuristics. We sorted the variables in decreasing order of information content and observed the decay in precision as the first variable, the first two variables, first three variables, etc. were removed from each model. The experiment showed that the heuristics precision drops abruptly (from 50%to 30%), while the peer-based prediction drops from 54% to 44% and then stabilizes at that level. The peers designed by the Evolutionary Algorithms) provided the best accuracy overall:

–60.3% = over 3 x random, 1.2x heuristics on selection for top 20% *(based on past performance)*

–63.5% = over 10 x random, 1.7 x heuristics on selection for top 52 units *(based on past performance)*

–55.0% = over 2.5 x random, 1.5 x heuristics on selection for top 20% *(based on future performance).*

Dynamic criteria for peer recognition (by evolving attribute weighting) demonstrated the ability to adapt to changing operational and maintenance environments. The

evolution of the search parameters $\{(a_i, b_i)\}$ provided an additional performance improvement reducing the number of null queries while improving the overall precision of the classifier. Table 1 shows the value of the weights and search parameter settings for time-slice 3, while Figure 6 illustrates the evolution of the weights over the three time slices.

TABLE II
WEIGHTS AND SEARCH PARAMETERS FOR TIME-SLICE 3

Index	Feature	Weight	a	Range (a)	b	Range (b)
1	RY_Rec/Yr	9.11	5.6	[0-6]	3.73	[0.5-5]
2	RY_Rec/100K_Miles	8.81	5.3	[0-8]	2.57	[0.5-5]
3	RY_Rec/100K_Engine_Hrs	7.35	35.9	[20-45]	2.67	[0.5-5]
4	RY_Rec_Count	5.69	8.5	[2.5-12]	3.37	[0.5-5]
5	RY_Rec/100K_Eng_Hrs_Move	4.08	10.1	[5-30]	2.75	[0.5-5]
6	Tot_Rec_Count	1.33	18.7	[3-50]	3.20	[0.5-5]
7	R_Rec_Count	0.87	1.0	[0.5-8]	3.16	[0.5-5]

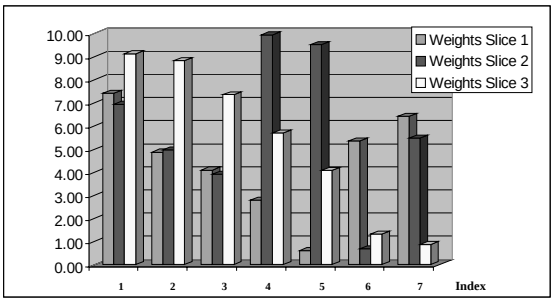

Fig. 6. Evolution of weights for the top seven variables over the three time slices (Index number described in Table I)

B. Future Directions

Our experiments have shown the applicability of evolutionary algorithms, used in a *wrapper* approach, to select the best attributes for *representing peers* and to define similarity measures for *identifying the most similar peers* for a given unit. By evolving the models over different time slices, we have also shown our ability to dynamically adapt the neighborhoods of peers using incremental operational and maintenance data. In future work, we could extend the structural design of the attribute space (for the definition of peers). By using genetic programming in lieu of evolutionary algorithms, we could extend *attribute selection* and *weighting* to *attribute construction*. We could also improve the fitness function to tradeoff classifier accuracy and confidence by adding measure of representation parsimony and find Pareto fronts for different tradeoffs.

We could also extend our approach by generating more sophisticated local models for prediction. Our current assumption was that each peer had a rather "feeble" track-history, which motivated the peer approach to begin with. As such we could not use too many degrees of freedom in creating the local models and we limited ourselves to an exponential average, completely defined by the value of α. In situations where the peers have a richer track-history, we could experiment with more complex models [10], whose parameters could be obtained using a local search method, e.g., error minimization, inside each EA's trial. This would be another example of intertwining local with global search [13].

By evolving the models over different time slices, we have shown a possible role that evolutionary search can play in the maintenance of models, as EA's provide a mechanism for model updating, preventing model obsolescence, and supporting model lifecycle [1,12].

ACKNOWLEDGMENT

This work was funded by DARPA, through contract CACI 621-04-S-0031. The authors gratefully acknowledge the help of many individuals and organizations that made this work possible, such as Drs. Norm Sondheimer, Al Wallace, and Peter Will, members of DARPA Steering Committee, and GE Rail who provided us with the data sets and domain knowledge that were indispensable for the generation of the models and the validation of the experiments.

REFERENCES

[1] P. Bonissone, "Development and maintenance of fuzzy models in financial applications" in *Soft Methodology and Random Information Systems*, M. Lopez-Diaz, M. Gil, P. Grzegorzwski, O. Hyrniewicz, J. Lawry, Eds. Berlin, Germany: Springer Verlag, 2004, pp. 50-66.

[2] D.B. Fogel, *Evolving Artificial Intelligence*, Ph.D. Dissertation, Univ. of California San Diego, CA, 1992.

[3] E.Vonk, L.Jain, and R. Johnson, *Automatic Generation of Neural Network Architecture Using Evolutionary Computation*. Singapore: World Scientific Pub., 1997.

[4] P. Larrañaga, J. Lozano, "Synergies between evolutionary computation and probabilistic graphical models", *Int. Journal of Approximate Reasoning*, Vol 31(3), 2002.

[5] C.L. Karr, "Design of an adaptive fuzzy logic controller using genetic algorithms", *Proc. Int. Conf. on Genetic Algorithms (ICGA'91)*, S. Diego, CA, 1991, pp. 450-456

[6] P. Bonissone, R. Subbu, and K. Aggour, "Evolutionary optimization of fuzzy decision systems for automated insurance underwriting," in *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems*, Honolulu. Piscataway, NJ: IEEE, 2002, pp. 1003-1008.

[7] H. Witten and E. Frank, *Data Mining: Practical machine learning tools with Java implementations*, San Francisco, CA: Morgan Kaufmann, 2000.

[8] C.G. Atkeson, "Memory-based approaches to approximating continuous functions", in *Nonlinear Modeling and Forecasting*, M. Casdagli and S. Eubank, Eds. Harlow, UK: Addison Wesley, 1992, pp. 503-521.

[9] H. Bersini, G. Bontempi, and M. Birattari, "Is readability compatible with accuracy? From neuro-fuzzy to lazy learning", *Proceedings in Artificial Intelligence 7*, C. Freksa, Ed. Berlin, Germany: Infix/Aka, 1998, pp. 10-25.

[10] C.G. Atkeson, A.W. Moore, S. Shaal, "Locally Weighted Learning" *Artificial Intelligence Review*, 11:11-73, 1997.

[11] A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Springer-Verlag, Berlin, 2002.

[12] P. Bonissone, "The life cycle of a fuzzy knowledge-based classifier", *Proceedings 2003 NAFIPS*, Chicago, IL. Piscataway, NJ: IEEE, 2003, pp. 488-494.

[13] J.-M Renders, H. Bersini, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways**",** *Proceedings 1st IEEE CEC*, Orlando, FL. NJ: IEEE, 1994. 312 - 317 vol.1.